Applied Genome Research

De novo transcriptome assembly

205048 & 205049

Boas Pucker

(https://github.com/trinityrnaseq/trinityrnaseq/wiki)

Generates initial assembly of dominant isoforms

Constructs graph of common sequences and unique sequences of different isoforms

Resolves graph and reports separate isoforms (final assembly)

Assembly requires trimming of reads

- Why is trimming required?
  - Removal of adapters to avoid artificial joins
- Start trimmomatic in SE mode on the RNA-Seq read sets!

Running Trinity

```
$ Trinity \
--normalize_reads \
--seqType fq \
--max_memory 20G \
--single <INPUT>.fastq \
--CPU 6 \
--output <OUTPUT_DIRECTORY>
```

Trinity on cluster

```
#!/bin/bash
echo "Trinity \
--normalize_reads \
--seqType fq \
--max_memory 10G \
--left 1_1P.fastq,2_1P.fastq \
--right 1_2P.fastq,2_2P.fastq \
--CPU 4 \
--output <SOME_DIRECTORY>" \
| qsub \
-cwd \
-N iGEM_trin \
-l vf=10G -l arch=lx-amd64 -l idle=1 \
-P fair_share \
-pe multislot 20 \
-o output.txt \
-e error.txt
```

Components of Trinity

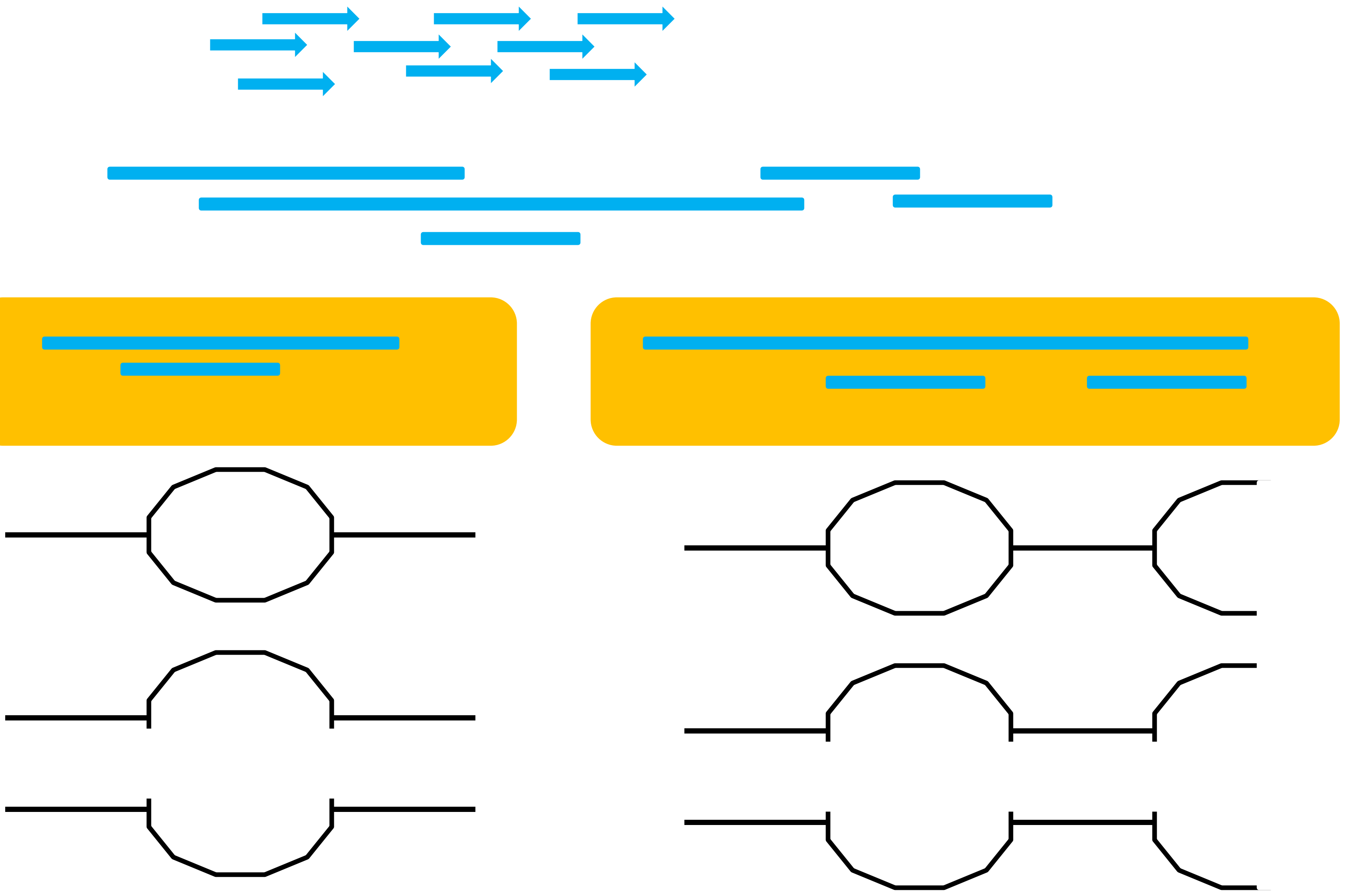
Illumina reads

Contigs

Contig clusters

de Bruijn graphs

Reconstructed isoforms



General analysis concept

Reads (per sample) → Combine reads

Normalization?

De novo assembly

Assembled transcripts (all samples)

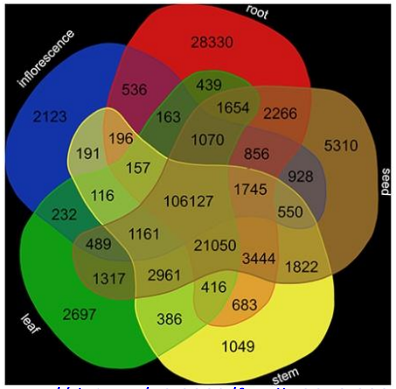
Abundance estimation

Identification of differentially expressed genes/transcripts

Expression patterns, transcript clusters

Assembled transcripts

Identify coding regions



https://doi.org/10.3389/fmolb.2018.00062

Best practice workflow

1) Generate resource report:

   trinityrnaseq-Trinity-v2.4.0/trinity-plugins/COLLECTL/examine_resource_usage_profiling.pl collect

2) Check assembly for full length transcripts => BLASTx vs. nr

3) Analyze tophit coverage

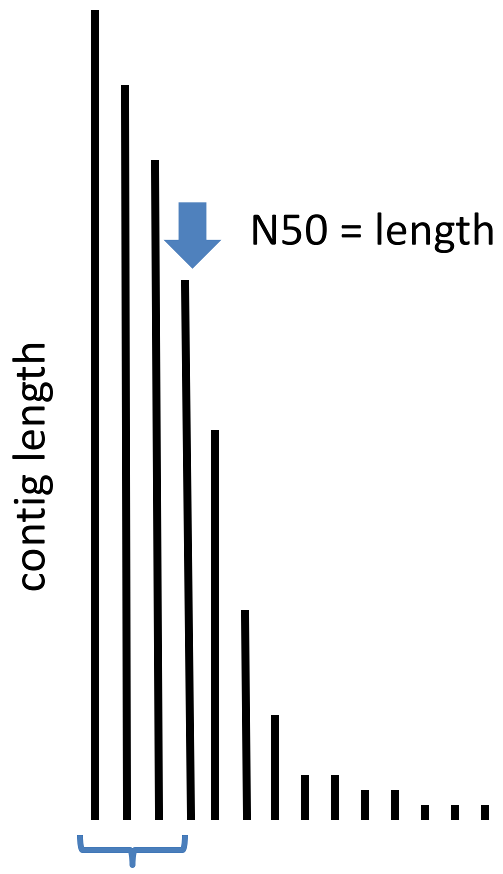
4) Check integrated hits via bowtie mapping against assembly

5) Calculate Nx stats e.g. Ex90N50 (N50 is not useful)

6) Run BUSCO to check assembly completeness

7) Run Interproscan to assign GO terms

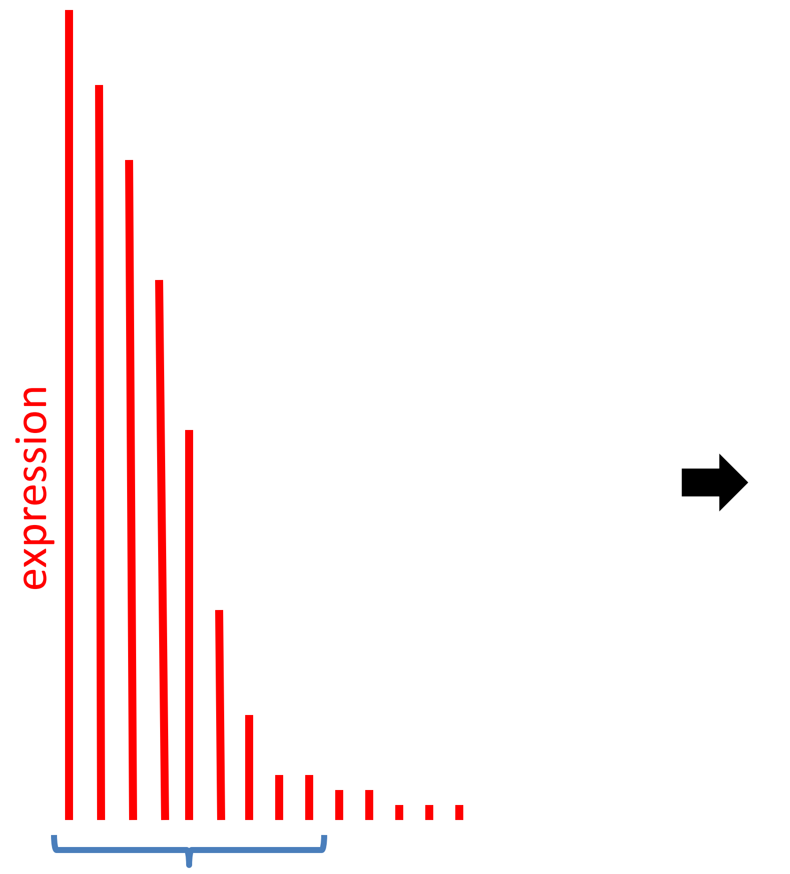
Assembly evaluation – Nx for continuity quantification



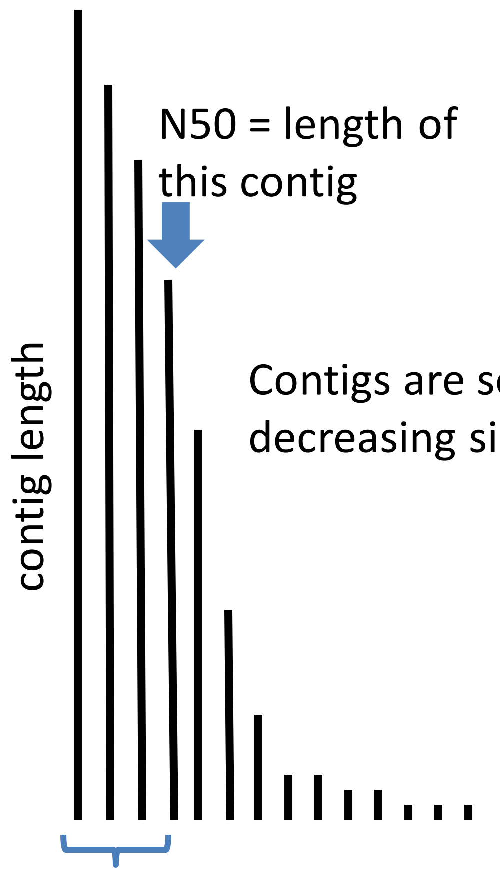
Contigs sum up to 50% of total assembly size

Contigs are sorted by decreasing size.

Assembly evaluation – Ex90N50

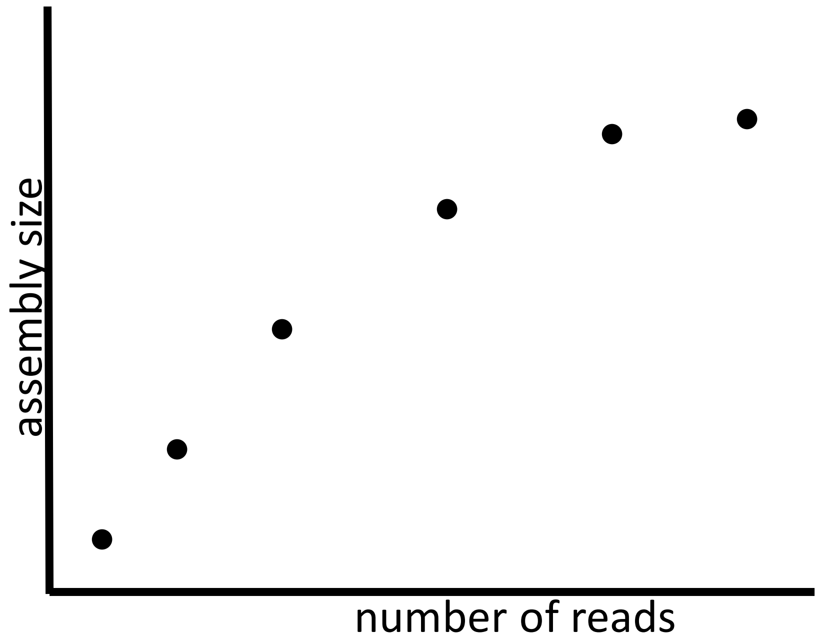


Sorting contigs by expression and selecting all sequences that account for 90% of all expression


contig length
N50 = length of this contig
Contigs are sorted by decreasing size.

Contigs sum up to 50% of selected assembly fraction

Saturation of assembly size



BUSCO

- “quantitative measures for the assessment of genome assembly, gene set, and transcriptome completeness, based on evolutionarily-informed expectations of gene content from near-universal single-copy orthologs ”
- https://busco.ezlab.org/
- Applications: assembly completeness assessment, estimation of heterozygosity, optimization of gene prediction, identification of paralogs, …

Gene Ontology (GO) terms

- http://geneontology.org/
- Computational representation of biological knowledge
- Over 40k biological concepts
- Used for automatic annotation of predicted genes
- GO enrichment analyses (e.g. in RNA-Seq studies)

Construct GFF3

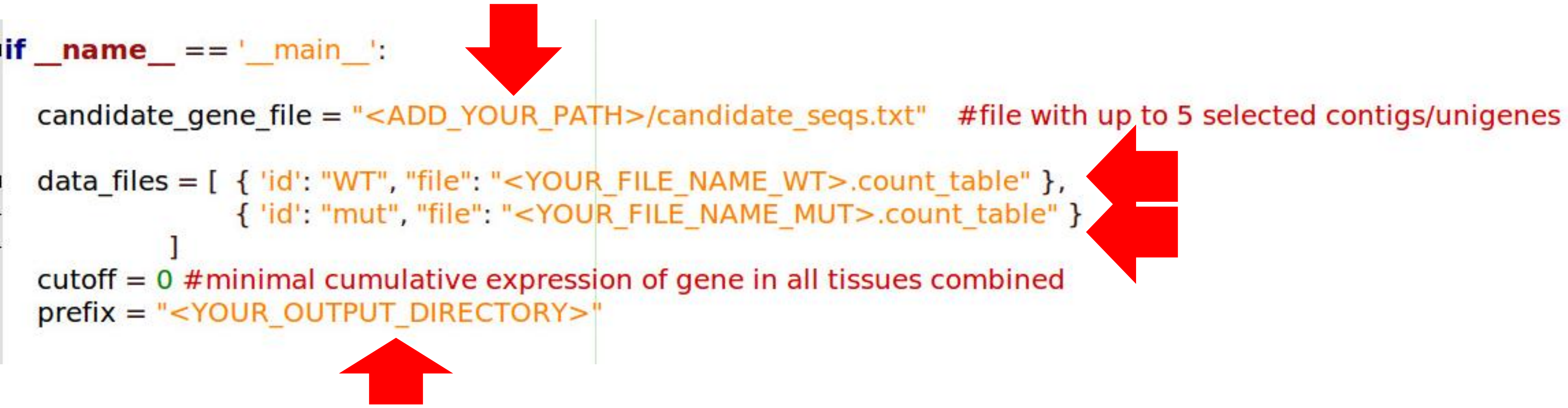
- One feature/entry in GFF3 file is created per sequence in FASTA file
- Length of sequences is used to determine start/end
- Running number is used to generate unique IDs

EXERCISE

1) Run ‘contig_stats.py’ on Trinity.fasta!
2) Use STAR to map all WT and 3xmyb reads to assembly!
3) Construct reference file for read counting via ‘fasta2gff.py’!
4) Use featureCounts to get expression values! (feature type = mRNA)
5) Construct heatmap via ‘construct_heatmap.py’! (see tipps)

Construct Heatmap

Add your own file paths and file names!



```
if __name__ == '__main__':

    candidate_gene_file = "<ADD_YOUR_PATH>/candidate_seqs.txt"    #file with up to 5 selected contigs/unigenes

    data_files = [  { 'id': "WT", "file": "<YOUR_FILE_NAME_WT>.count_table" },
                    { 'id': "mut", "file": "<YOUR_FILE_NAME_MUT>.count_table" }
                ]
    cutoff = 0 #minimal cumulative expression of gene in all tissues combined
    prefix = "<YOUR_OUTPUT_DIRECTORY>"
```

Run script like always: python construct_heatmap.py

EXERCISE

- Identify MYB11, MYB12, and MYB111 via BLAST!
- Is there expression different between samples?